Parabolic Partial Differential Equations

1. Partial Differential Equations
   - the heat equation
2. Forward Differences
   - discretization of space and time
   - time stepping formulas
   - stability analysis
3. Backward Differences
   - unconditional stability
   - the Crank-Nicholson method

MCS 471 Lecture 38
Numerical Analysis
Jan Verschelde, 18 November 2022

Parabolic Partial Differential Equations

1. Partial Differential Equations
   - the heat equation
2. Forward Differences
   - discretization of space and time
   - time stepping formulas
   - stability analysis
3. Backward Differences
   - unconditional stability
   - the Crank-Nicholson method

Partial Differential Equations

We look for functions $u(x, y)$ as the solution of the Partial Differential Equation (or PDE):

$$Au_{xx} + Bu_{xy} + Cu_{yy} + F(x, y, u, u_x, u_y) = 0,$$

where x and y are the independent variables, and

$$u_x = \frac{du}{dx}, \quad u_y = \frac{du}{dy}, \quad u_{xx} = \frac{d^2u}{dx^2}, \quad u_{yy} = \frac{d^2u}{dy^2}, \quad u_{xy} = \frac{d^2u}{dxdy}.$$

Instead of y, time t may be the second independent variable.

Second order PDEs are classified by $B^2 - 4AC$:

- If $B^2 - 4AC = 0$, then the PDE is *parabolic* (heat).
- If $B^2 - 4AC > 0$, then the PDE is *hyperbolic* (wave).
- If $B^2 - 4AC < 0$, then the PDE is *elliptic* (steady state).

the heat equation

Consider:

- The temperature is measured in an interval from a to b.
- The independent variables are $x \in [a, b]$ and time $t \geq 0$.
- The diffusion coefficient is $D > 0$.

The heat equation comes from materials science (Fick's second law).

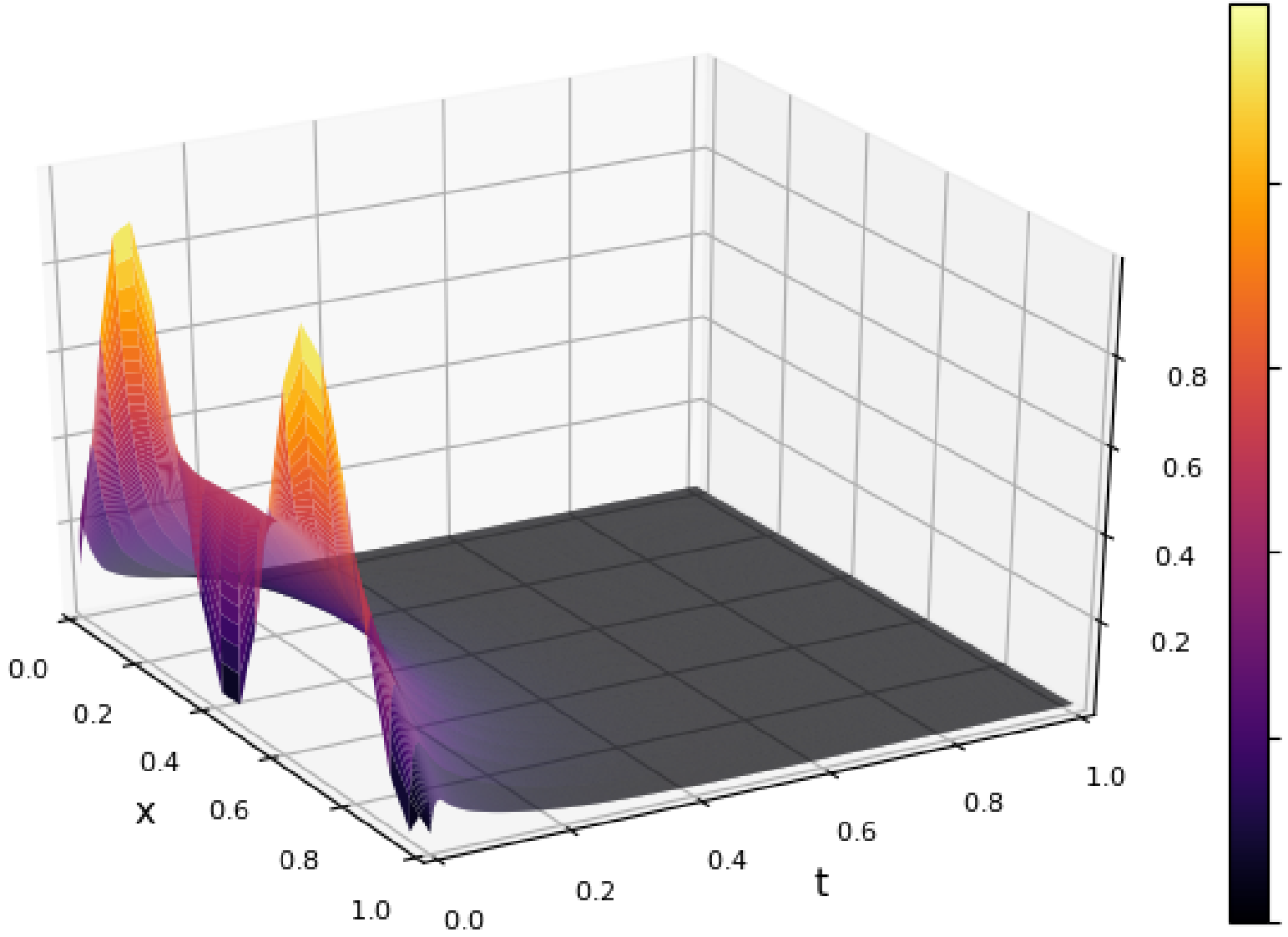
The *heat equation* has the format

$$\begin{cases} u_t = Du_{xx} & \text{for all } x \in [a,b], t \geq 0, \\ u(x,0) = f(x) & \text{for all } x \in [a,b], \\ u(a,t) = \ell(t) & \text{for all } t \geq 0, \\ u(b,t) = r(t) & \text{for all } t \geq 0. \end{cases}$$

We have the initial temperature distribution given by $f(x)$.

The functions $\ell(t)$ and $r(t)$ give the temperature at the ends.

$D = 1$, $f(x) = \sin^2(2\pi x)$, $\ell(t) = 0 = r(t)$

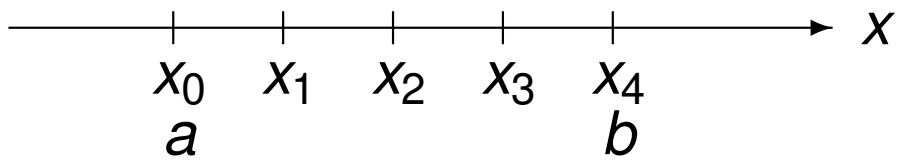


Parabolic Partial Differential Equations

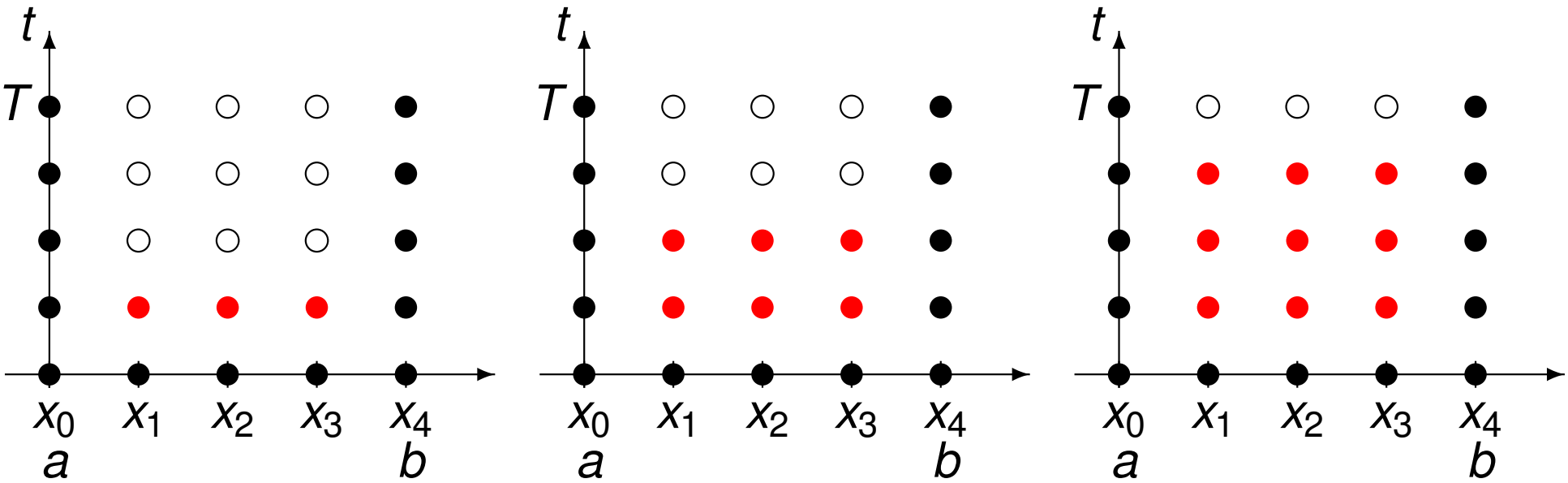
1. Partial Differential Equations
   - the heat equation
2. Forward Differences
   - discretization of space and time
   - time stepping formulas
   - stability analysis
3. Backward Differences
   - unconditional stability
   - the Crank-Nicholson method

discretization of space and time

We discretize in space, $x \in [a, b]$:

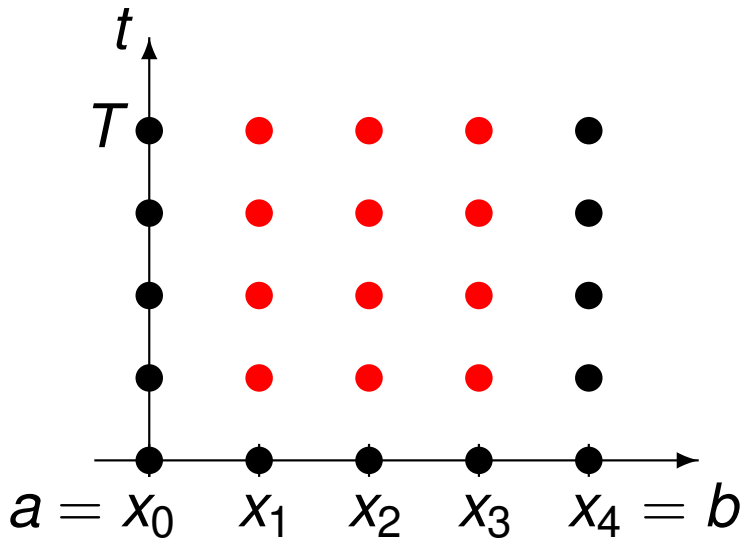


and in time, $t \in [0, T]$:



forward differences in time

- Let M be the number of steps in space, $h = (b - a)/M$.
- Let N be the number of steps in time, $k = T/N$.

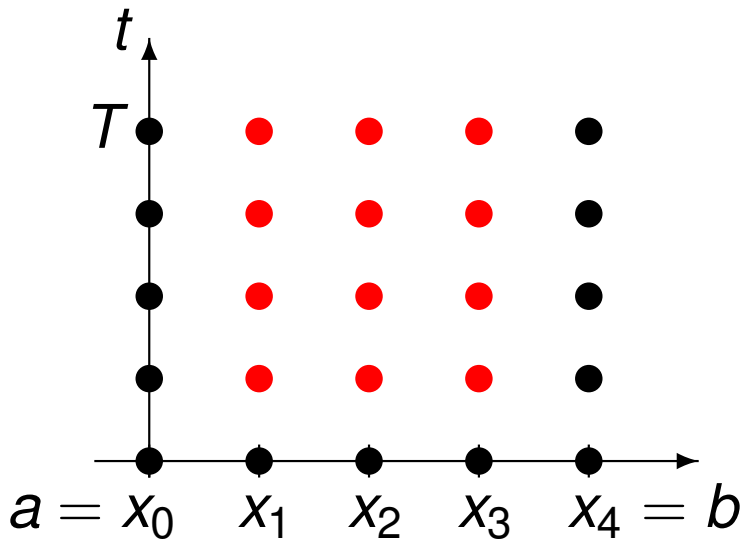


Applying the forward difference formula in time:

$$u_t(x,t) = \frac{u(x,t+k) - u(x,t)}{k} + \frac{k}{2} u_{tt}(x,c_2), \quad t < c_2 < t + h.$$

central differences in space

- Let M be the number of steps in space, $h = (b - a)/M$.
- Let N be the number of steps in time, $k = T/N$.



Applying the central difference formula in space:

$$u_{xx}(x,t) = \frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{h^2} + \frac{h^2}{12} u_{xxxx}(c_1,t),$$

for $x - h < c_1 < x + h$.

Parabolic Partial Differential Equations

1. Partial Differential Equations
   - the heat equation
2. Forward Differences
   - discretization of space and time
   - time stepping formulas
   - stability analysis
3. Backward Differences
   - unconditional stability
   - the Crank-Nicholson method

time stepping formulas

Substituting the formulas into the heat equation $u_t = Du_{xx}$:

$$D\frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{h^2} = \frac{u(x,t+k) - u(x,t)}{k}$$

which has local truncation errors of $O(h^2)$ and $O(k)$.

To solve the differential equation, apply *time stepping formulas*:

$$\begin{aligned} y_{i,j+1} &= y_{i,j} + \frac{Dk}{h^2}\left(y_{i+1,j} - 2y_{i,j} + y_{i-1,j}\right), \quad \text{at } (x_i, t_j, y_{i,j}) \\ &= \sigma y_{i+1,j} + (1-2\sigma)y_{i,j} + \sigma y_{i-1,j}, \quad \sigma = Dk/h^2. \end{aligned}$$

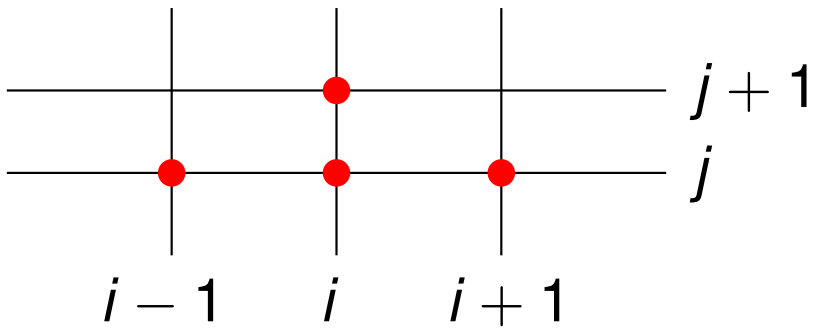
The time stepping methods are *explicit*,
what is a the right of the equation for $y_{i,j+1}$ is known.

the stencil of the method

For $i = 1, 2, \ldots, M-1$ and $j = 0, 1, \ldots, T$:

$$y_{i,j+1} = \sigma y_{i+1,j} + (1 - 2\sigma) y_{i,j} + \sigma y_{i-1,j}.$$

The *stencil of the method* defines the set of the involved points.



The stencil shows that the method is explicit.

the matrix formula

For $i = 1, 2, \ldots, M-1$ and $j = 0, 1, \ldots, T$:

$$y_{i,j+1} = \sigma y_{i+1,j} + (1 - 2\sigma) y_{i,j} + \sigma y_{i-1,j}.$$

$$\begin{bmatrix} y_{1,j+1} \\ y_{2,j+1} \\ \vdots \\ y_{M-2,j+1} \\ y_{M-1,j+1} \end{bmatrix} = \begin{bmatrix} 1-2\sigma & \sigma & & & \\ \sigma & 1-2\sigma & \sigma & & \\ & \ddots & \ddots & \ddots & \\ & & \sigma & 1-2\sigma & \sigma \\ & & & \sigma & 1-2\sigma \end{bmatrix} \begin{bmatrix} y_{1,j} \\ y_{2,j} \\ \vdots \\ y_{M-2,j} \\ y_{M-1,j} \end{bmatrix}$$

$$+ \sigma \begin{bmatrix} y_{0,j} \\ 0 \\ \vdots \\ 0 \\ y_{M,j} \end{bmatrix}$$

specification of a Julia function

```
"""
    heatforward(D::Float64,f0::Function,lb::Function,rb::Function,
                a::Float64,b::Float64,T::Float64,M::Int,N::Int)

returns the M-by-N matrix of approximated values over [a,b]
with time running from 0 to T.

The diffusion coefficient is given by D.

f0 gives the initial temperature distribution over [a,b],

lb defines the temperature at the left bound of [a,b],

rb defines the temperature at the right bound of [a,b].

EXAMPLE :
    f(x) = (sin(2*pi*x))^2
    lb(t) = 0
    rb(t) = 0
    sol = heatforward(1.0,f,lb,rb,0.0,1.0,1.0,10,250)
"""
```

definition of the Julia function

```
function heatforward(D::Float64,f0::Function,lb::Function,
                     rb::Function,a::Float64,b::Float64,
                     T::Float64,M::Int,N::Int)
    result = zeros(M,N)
    offset = zeros(M)
    h = (b-a)/(M-1)
    k = T/(N-1)
    sigma = (D*k)/h^2
    Adiag = (1-2*sigma)*ones(M)
    Asubd = sigma*ones(M-1)
    A = diagm(Adiag) + diagm(+1 => Asubd) + diagm(-1 => Asubd)
    result[:,1] = Array([f0(i*h) for i=1:M])
    for j=2:N
        offset[1] = sigma*lb((j-1)*k)
        offset[M] = sigma*rb((j-1)*k)
        result[:,j] = A*result[:,j-1] + offset
    end
    return result
end
```

running for $M = 6$ and $N = 6$

```
The solution :
6×6 Array{Float64,2}:
 9.045e-01  -6.413e+00   7.342e+01  -8.839e+02   1.068e+04  -1.235e+05
 3.455e-01   3.141e+00  -4.463e+01   5.456e+02  -5.471e+03   3.003e+04
 3.455e-01   3.141e+00  -4.463e+01   7.717e+02  -1.453e+04   2.798e+05
 9.045e-01  -6.413e+00   1.186e+02  -2.062e+03   3.528e+04  -6.034e+05
 5.999e-32   9.045e+00  -1.542e+02   2.573e+03  -4.265e+04   7.073e+05
 9.045e-01  -8.141e+00   1.185e+02  -1.837e+03   2.940e+04  -4.779e+05
```

The rows are indexed by space, the columns advance in time.

Observe the values in the rightmost column: divergent!

For the plot: $M = 20$ and $N = 1000$.

Why $N \gg M$ for correctness?

Parabolic Partial Differential Equations

1. Partial Differential Equations
   - the heat equation
2. Forward Differences
   - discretization of space and time
   - time stepping formulas
   - stability analysis
3. Backward Differences
   - unconditional stability
   - the Crank-Nicholson method

stability analysis of the forward difference method

The forward difference method applies the formula

$$\mathbf{y}_{j+1} = A\mathbf{y}_j + \mathbf{s}_j, \quad A \text{ is tridiagonal}, \quad j = 0, 1, \ldots, N.$$

The exact solution also satisfies $\mathbf{u}_{j+1} = A\mathbf{u}_j + \mathbf{s}_j$.

The error $\mathbf{e}_{j+1}$ is then

$$\begin{aligned} \mathbf{e}_{j+1} &= \mathbf{y}_{j+1} - \mathbf{u}_{j+1} \\ &= A\mathbf{y}_j + \mathbf{s}_j - (A\mathbf{u}_j + \mathbf{s}_j) \\ &= A(\mathbf{y}_j - \mathbf{u}_j) \\ &= A\mathbf{e}_j = A^2\mathbf{e}_{j-1} = \cdots = A^{j+1}\mathbf{e}_0. \end{aligned}$$

For the error $\mathbf{e}_{j+1}$ to go to zero, we require: $\|A\|_2 < 1$.

$\|A\|_2$ is the magnitude of the largest eigenvalue of A.

A method is *stable* if the error goes to zero.

some numerical examples

$$A = \begin{bmatrix} 1-2\sigma & \sigma & & & \\ \sigma & 1-2\sigma & \sigma & & \\ & \ddots & \ddots & \ddots & \\ & & \sigma & 1-2\sigma & \sigma \\ & & & \sigma & 1-2\sigma \end{bmatrix}, \quad \sigma = Dk/h^2,$$

where $h = (b-a)/M$ and $k = T/N$.

Some numerical examples:

- For $M = 6$ and $N = 6$, $\|A\|_2 = 1.801\text{e+}01$.
- For $M = 10$ and $N = 250$, $\|A\|_2 = 9.736\text{e-}01$.
- For $M = 20$ and $N = 1000$, $\|A\|_2 = 9.919\text{e-}01$.

Von Neumann stability analysis

Theorem

Consider the heat equation over $x \in [a, b]$, for $t \in [0, T]$, $D > 0$,

$$\text{with } h = (b-a)/M \text{ and } k = T/N.$$

If $\dfrac{Dk}{h^2} < \dfrac{1}{2}$, then the forward difference method is stable.

Some numerical examples ($D = 1$):

- For $M = 6$ and $N = 6$, $h = 1/6 = k$, $Dk/h^2 = 6$.
- For $M = 10$ and $N = 250$, $h = 1/10$, $k = 1/250$, $Dk/h^2 = 0.4$.
- For $M = 20$ and $N = 1000$, $h = 1/20$, $k = 1/1000$, $Dk/h^2 = 0.4$.

The explicit forward difference method is *conditionally stable*.

find the critical threshold step size

Exercise 1: Consider the example of the heat equation

$$\begin{cases} u_t &= u_{xx} & \text{for all } x \in [0,1], t \in [0,1], \\ u(x,0) &= \sin^2(2\pi x) & \text{for all } x \in [0,1], \\ u(a,t) &= 0 & \text{for all } t \in [0,1], \\ u(b,t) &= 0 & \text{for all } t \in [0,1]. \end{cases}$$

1. For $M = 20$, what is the critical threshold on N for the method to be stable?

   What is the smallest value for N (or the largest value for k) so the method is stable? Justify your answer.

2. Verify your bound experimentally.

Parabolic Partial Differential Equations

1. Partial Differential Equations
   - the heat equation
2. Forward Differences
   - discretization of space and time
   - time stepping formulas
   - stability analysis
3. Backward Differences
   - unconditional stability
   - the Crank-Nicholson method

using backward differences in time

Recalling the backward Euler method, to avoid the error magnification problems, we apply backward differences in time:

$$u_t(x,t) = \frac{u(x,t) - u(x,t-k)}{k} + \frac{k}{2} u_{tt}(x,c_0), \quad t-k < c_0 < t.$$

In space, as before, we apply central differences.

$$\frac{1}{k}\left(y_{i,j} - y_{i,j-1}\right) = \frac{D}{h^2}\left(y_{i+1,j} - 2y_{i,j} + y_{i-1,j}\right)$$

with local truncation error $O(k) + O(h^2)$.

Collecting all values indexed by j to the left gives:

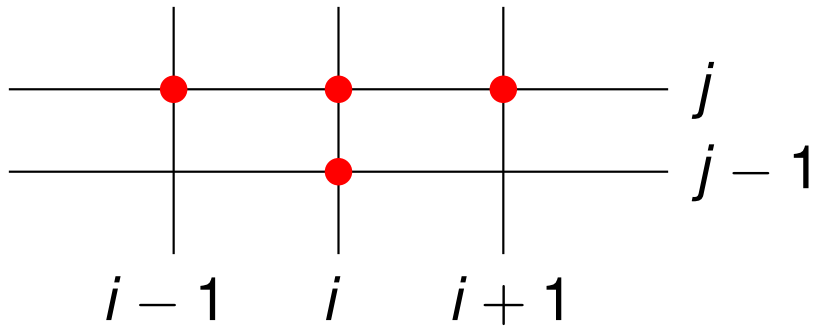
$$-\frac{Dk}{h^2} y_{i+1,j} + \left(1 + 2\frac{Dk}{h^2}\right) y_{i,j} - \frac{Dk}{h^2} y_{i-1,j} \quad = \quad y_{i,j-1}.$$

the backward difference method

As before, set $\sigma = Dk/h^2$:

$$-\sigma y_{i+1,j} + (1+2\sigma)\, y_{i,j} - \sigma y_{i-1,j} \;=\; y_{i,j-1}.$$

The stencil of the method:



shows that the backward difference method is *implicit*.

the matrix formula

$$\begin{bmatrix} 1+2\sigma & -\sigma & & & \\ -\sigma & 1+2\sigma & -\sigma & & \\ & \ddots & \ddots & \ddots & \\ & & -\sigma & 1+2\sigma & -\sigma \\ & & & -\sigma & 1+2\sigma \end{bmatrix} \begin{bmatrix} y_{1,j} \\ y_{2,j} \\ \vdots \\ y_{M-2,j} \\ y_{M-1,j} \end{bmatrix} = \begin{bmatrix} y_{1,j-1} \\ y_{2,j-1} \\ \vdots \\ y_{M-2,j-1} \\ y_{M-1,j-1} \end{bmatrix}$$

$$+ \sigma \begin{bmatrix} y_{0,j} \\ 0 \\ \vdots \\ 0 \\ y_{M,j} \end{bmatrix}$$

In short: $\mathbf{y}_j = A^{-1}\mathbf{y}_{j-1} + \mathbf{b}$.

the function `heatbackward`

Modifications to the `heatforward` function,
in the definition of the function `heatbackward`:

```
Adiag = (1+2*sigma)*ones(M)
Asubd = -sigma*ones(M-1)
A = diagm(Adiag)
  + diagm(+1 => Asubd) + diagm(-1 => Asubd)
result[:,1] = Array([f0(i*h) for i=1:M])
for j=2:N
    offset[1] = sigma*lb((j-1)*k)
    offset[M] = sigma*rb((j-1)*k)
    rhs = result[:,j-1] + offset
    result[:,j] = A\rhs
end
```

Observe the `A\rhs` for the formula $\mathbf{y}_j = A^{-1}\mathbf{y}_{j-1} + \mathbf{b}$.

running for $M = 6$ and $N = 6$

```
The solution :
6×6 Array{Float64,2}:
 9.045e-01  1.978e-01  7.689e-02  3.642e-02  1.805e-02  9.039e-03
 3.455e-01  2.542e-01  1.296e-01  6.475e-02  3.243e-02  1.627e-02
 3.455e-01  2.923e-01  1.574e-01  8.011e-02  4.035e-02  2.028e-02
 9.045e-01  3.198e-01  1.583e-01  8.001e-02  4.031e-02  2.027e-02
 5.999e-32  2.303e-01  1.268e-01  6.425e-02  3.233e-02  1.625e-02
 9.045e-01  1.869e-01  7.463e-02  3.599e-02  1.797e-02  9.021e-03
```

Although the h and k values are too large,

the numbers are no longer diverging as with forward differences.

running on an example

Exercise 2: Consider the example of the heat equation

$$\begin{cases} u_t &= u_{xx} & \text{for all } x \in [0,1], t \in [0,1], \\ u(x,0) &= \sin^2(2\pi x) & \text{for all } x \in [0,1], \\ u(a,t) &= 0 & \text{for all } t \in [0,1], \\ u(b,t) &= 0 & \text{for all } t \in [0,1]. \end{cases}$$

Apply the backward difference method on this problem.

Let $M = 20$ and find a good value for N which provides the same accuracy as the forward difference method with $N = 1000$.

Parabolic Partial Differential Equations

1. Partial Differential Equations
   - the heat equation
2. Forward Differences
   - discretization of space and time
   - time stepping formulas
   - stability analysis
3. Backward Differences
   - unconditional stability
   - the Crank-Nicholson method

mixed differences in space

For u_{xx}, apply the weighted combination

$$\frac{1}{2}\left(\frac{y_{i+1,j} - 2y_{i,j} + y_{i-1,j}}{h^2}\right) + \frac{1}{2}\left(\frac{y_{i+1,j-1} - 2y_{i,j-1} + y_{i-1,j-1}}{h^2}\right).$$

For u_t, use $\frac{1}{k}\left(y_{i,j} - y_{i,j-1}\right)$. Set $\sigma = Dk/h^2$.

We then obtain

$$2y_{i,j} - 2y_{i,j-1} = \sigma\left(y_{i+1,j} - 2y_{i,j} + y_{i-1,j} + y_{i+1,j-1} - 2y_{i,j-1} + y_{i-1,j-1}\right),$$

or equivalently:

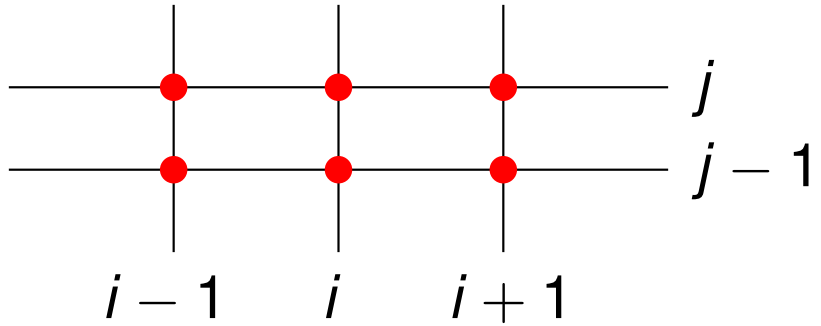
$$-\sigma y_{i-1,j} + (2+2\sigma)y_{i,j} - \sigma y_{i+1,j} = \sigma y_{i-1,j-1} + (2-2\sigma)y_{i,j-1} + \sigma y_{i+1,j-1}.$$

the stencil of the method

The local error is $O(h^2) + O(k^2)$.

$$-\sigma y_{i-1,j} + (2+2\sigma)y_{i,j} - \sigma y_{i+1,j} = \sigma y_{i-1,j-1} + (2-2\sigma)y_{i,j-1} + \sigma y_{i+1,j-1}.$$

The stencil of the method:



In matrix form, the Crank-Nicholson method is

$$A\mathbf{y}_j = B\mathbf{y}_{j-1} + \sigma(\mathbf{s}_{j-1} + \mathbf{s}_j), \quad j = 1, 2, \ldots, N,$$

where $\mathbf{s}_j = [y_{0,j}, 0, \ldots, 0, y_{M,j}]^T$.

The method is unconditionally stable.

the function `heatcrank`

Modifications to the function `heatbackward`:

```
Adiag = (2+2*sigma)*ones(M)
Asubd = -sigma*ones(M-1)
A = diagm(Adiag)
  + diagm(+1 => Asubd) + diagm(-1 => Asubd)
Bdiag = (2-2*sigma)*ones(M)
B = diagm(Bdiag)
  + diagm(+1 => -Asubd) + diagm(-1 => -Asubd)
result[:,1] = Array([f0(i*h) for i=1:M])
for j=2:N
    offset[1] = sigma*(lb((j-1)*k) + lb(j*k))
    offset[M] = sigma*(rb((j-1)*k) + rb(j*k))
    rhs = B*result[:,j-1] + offset
    result[:,j] = A\rhs
end
```

running for $M = 6$ and $N = 6$

```
The solution :
6×6 Matrix{Float64}:
 9.045e-01 -3.235e-01  3.207e-01 -1.918e-01  1.479e-01 -1.027e-01
 3.455e-01  3.253e-01 -7.314e-02  1.261e-01 -7.787e-02  6.811e-02
 3.455e-01  4.070e-01 -5.929e-02  1.160e-01 -5.724e-02  5.087e-02
 9.045e-01 -4.571e-02  3.025e-01 -1.720e-01  1.719e-01 -1.316e-01
 5.999e-32  5.850e-01 -2.800e-01  2.944e-01 -2.158e-01  1.813e-01
 9.045e-01 -3.592e-01  3.666e-01 -2.384e-01  1.917e-01 -1.422e-01
```

The truncation error of backward difference method: $O(k) + O(h^2)$.

The truncation error of the Crank-Nicholson method: $O(k^2) + O(h^2)$.

running on an example

Exercise 3: Consider the example of the heat equation

$$\begin{cases} u_t &= u_{xx} & \text{for all } x \in [0,1], t \in [0,1], \\ u(x,0) &= \sin^2(2\pi x) & \text{for all } x \in [0,1], \\ u(a,t) &= 0 & \text{for all } t \in [0,1], \\ u(b,t) &= 0 & \text{for all } t \in [0,1]. \end{cases}$$

Apply the Crank-Nicholson method on this problem.

Let $M = 20$ and find a good value for N which provides the same accuracy as the forward difference method with $N = 1000$.